

PAT-NO: JP409231093A
DOCUMENT-IDENTIFIER: JP 09231093 A
TITLE: METHOD AND SYSTEM FOR TRANSFER OF PROGRAM CONTROL
BETWEEN TWO ARCHITECTURES
PUBN-DATE: September 5, 1997

INVENTOR-INFORMATION:

NAME	COUNTRY
JOHN, W GOETTSU	
JOHN, M KETTY	
STEFAN, W MANN	

ASSIGNEE-INFORMATION:

NAME	COUNTRY
INTERNATL BUSINESS MACH CORP	N/A

APPL-NO: JP09024598
APPL-DATE: February 7, 1997

INT-CL (IPC): G06F009/46 , G06F009/455

ABSTRACT:

PROBLEM TO BE SOLVED: To provide a computer system which supports the transfer of control between two architectures in different methods for the call programs including the transfer of different word lengths and/or parameters, stack pointers and return addresses.

SOLUTION: The program control is transferred between two architectures in a control transfer mode bit. A bookkeeping system is generated in consideration of the call and return instructions between both architectures and holds the data received from the calling architecture to attain the different word length between the architectures. Then the stack pointers and return address of the master and slave call programs are stored in a register not in a main memory.

COPYRIGHT: (C) 1997, JPO

(19)日本国特許庁 (J P)

(12) 公 開 特 許 公 報 (A)

(11)特許出願公開番号

特開平9-231093

(43)公開日 平成9年(1997)9月5日

(51)Int.Cl. ⁶	識別記号	庁内整理番号	F I	技術表示箇所	
G 0 6 F	9/46	3 6 0	G 0 6 F	9/46	3 6 0 B
	9/455			9/44	3 1 0 A

審査請求 未請求 請求項の数38 O L (全 14 頁)

(21)出願番号	特願平9-24598	(71)出願人	390009531 インターナショナル・ビジネス・マシーンズ・コーポレーション INTERNATIONAL BUSIN ESS MASCHINES CORPO RATION アメリカ合衆国10504、ニューヨーク州 アーモンク (番地なし)
(22)出願日	平成9年(1997)2月7日	(72)発明者	ジョン・ダブリュ・ゴエツ アメリカ合衆国05465、バーモント州ジェ リコ、バターカップ・レーン 3
(31)優先権主張番号	0 8 / 6 0 5 4 0 9	(74)代理人	弁理士 合田 潔 (外2名)
(32)優先日	1996年2月22日		
(33)優先権主張国	米国 (U S)		

最終頁に続く

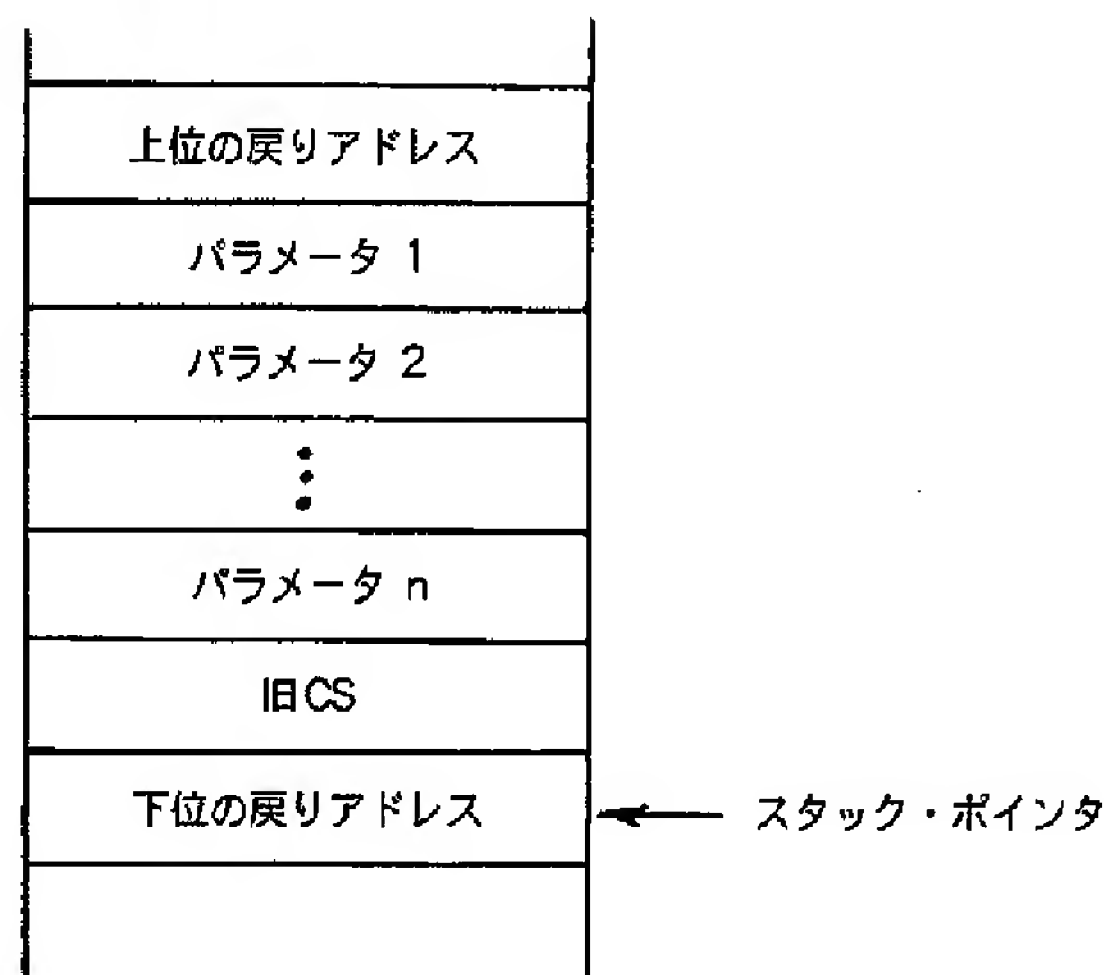
(54)【発明の名称】 2つのアーキテクチャ間でプログラム制御を転送する方法及びシステム

(57)【要約】

【課題】 異なるワード長及び(または)、パラメータ、スタック・ポインタ及び戻りアドレスの引き渡しを含む呼び出しプログラムにおいて、異なる方法で2つのアーキテクチャ間の制御転送をサポートするコンピュータ・システムを提供する。

【解決手段】 プログラム制御は制御転送モード・ビットによってアーキテクチャ間で転送される。ブックキーピング方式が2つのアーキテクチャ間の呼び出し及び戻りの命令を考慮して生成され、呼び出しアーキテクチャからのデータを保持し、アーキテクチャ間のワード長の違いを可能にする。マスタ/スレーブ関係の呼び出しプログラムのスタック・ポインタ及び戻りアドレスを、メイン・メモリではなくレジスタに格納する。

呼び出し後の
新たなスタック



【特許請求の範囲】

【請求項1】 a) 第1のマイクロプロセッサ・アーキテクチャと、

b) 前記第1のマイクロプロセッサ・アーキテクチャとは異なる第2のマイクロプロセッサ・アーキテクチャと、

c) 前記第1及び第2のマイクロプロセッサ・アーキテクチャの、プログラム制御を解放する一方から獲得する他方に該プログラム制御を転送する手段と、

を含み、前記プログラム制御を転送する手段が、該プログラム制御を解放するマイクロプロセッサ・アーキテクチャから情報を受け取り、該プログラム制御を獲得するマイクロプロセッサ・アーキテクチャに前記情報を提供する、マイクロプロセッサ。

【請求項2】 前記プログラム制御を転送する手段が、前記プログラム制御を獲得する前記マイクロプロセッサ・アーキテクチャが前記プログラム制御を解放する前記マイクロプロセッサ・アーキテクチャからのデータをメモリに必要としない場合、レジスタに該データを保持する手段を含む、請求項1記載のマイクロプロセッサ。

【請求項3】 前記データを保持する手段が、前記プログラム制御を解放する前記マイクロプロセッサ・アーキテクチャの基底アドレスのオフセットを含む、請求項2記載のマイクロプロセッサ。

【請求項4】 前記第2のマイクロプロセッサ・アーキテクチャが、前記第1のマイクロプロセッサ・アーキテクチャをエミュレートする手段を更に含む、請求項1記載のマイクロプロセッサ。

【請求項5】 前記プログラム制御を転送する手段は、前記第1及び第2のマイクロプロセッサ・アーキテクチャのどちらが前記プログラム制御を有するかを示す情報フィールドを含む、請求項1記載のマイクロプロセッサ。

【請求項6】 外部メモリにある前記情報フィールドが、バスによって前記マイクロプロセッサに転送される、請求項5記載のマイクロプロセッサ。

【請求項7】 前記情報フィールドが情報の1ビットを含み、前記ビットの状態が、前記第1及び第2のマイクロプロセッサ・アーキテクチャのどちらが前記プログラム制御を有するかを示す、請求項5記載のマイクロプロセッサ。

【請求項8】 前記第2のマイクロプロセッサ・アーキテクチャのアドレス範囲が、前記第1のマイクロプロセッサ・アーキテクチャのアドレス範囲より大きい、請求項1記載のマイクロプロセッサ。

【請求項9】 前記第1のマイクロプロセッサ・アーキテクチャはX86 CISCアーキテクチャである、請求項1記載のマイクロプロセッサ。

【請求項10】 前記第2のマイクロプロセッサ・アーキテクチャはPowerPC RISCアーキテクチャである、請求項1記載のマイクロプロセッサ。

【請求項11】 第1及び第2のプロセッサ・アーキテクチャ間でプログラム制御を転送する方法であって、

a) 前記第1のプロセッサ・アーキテクチャに対応する第1のプログラムを与えるステップと、

b) 前記第2のプロセッサ・アーキテクチャに対応する第2のプログラムを与えるステップと、

c) 前記第1及び第2のプロセッサ・アーキテクチャを示す複数の状態を有する制御転送モード手段を与えるステップと、

d) 次に実行される命令を有する前記プログラムに対応するプロセッサ・アーキテクチャを示す前記複数の状態の1つに、前記制御転送モード手段を設定するステップと、

e) 制御転送データをメイン・メモリまたはハードウェア・レジスタのどちらかに送信するかを決めるために、前記制御転送モード手段の状態を使用するステップと、

f) 前記制御転送モード手段の状態に従って、前記第1及び第2のプログラム間で制御転送データを送信するステップと、

を有する方法。

【請求項12】 前記制御転送モード手段は、情報の単一ビットの値にもとづいて前記第1及び第2のプロセッサ・アーキテクチャを示す、請求項11記載の方法。

【請求項13】 前記第1のプロセッサ・アーキテクチャのアドレス範囲が前記第2のプロセッサ・アーキテクチャのアドレス範囲より小さい、請求項11記載の方法。

【請求項14】 マイクロプロセッサ、及び外部バスで前記マイクロプロセッサに接続されている外部メモリを有するマイクロプロセッサ・システムの第1及び第2のプロセッサ・アーキテクチャ間にてプログラム制御を転送する方法であって、前記マイクロプロセッサは、前記外部メモリ、レジスタ及び実行ユニット間で接続されるバス及びキャッシュ・ユニットを有し、前記マイクロプロセッサは、自身が操作する前記レジスタにプロセス・スタックを格納するように構成され、前記方法は、

a) 制御転送モード手段にて、前記第2のプロセッサ・アーキテクチャから前記第1のプロセッサ・アーキテクチャへの前記プログラム制御の転送を示すステップと、

b) 保持手段において前記第2のプロセッサ・アーキテクチャに関するデータを保持するステップと、

c) 前記プログラム制御を前記第2のプロセッサ・アーキテクチャから前記第1のプロセッサ・アーキテクチャに転送するステップと、

d) 前記第2のプロセッサ・アーキテクチャに関するデータを、前記プログラム制御の転送の際に前記保持手段から前記第2のプロセッサ・アーキテクチャに戻すステップと、

を含む方法。

【請求項15】 e) 前記制御転送モード手段を、前記外部バスを介して前記外部メモリに適用するステップと、

を含む、請求項14記載の方法。

【請求項16】前記プログラム制御を前記第2のプロセッサ・アーキテクチャから前記第1のプロセッサ・アーキテクチャに転送するステップが、

c1) プログラムのアーキテクチャに従って、前記制御転送モード手段の状態を設定するステップと、

c2) 制御転送データが外部メモリまたはハードウェア・レジスタのどちらに送信されるかを定めるために前記制御転送モード手段の状態を使用するステップと、

c3) 前記制御転送手段の前記状態に従って、プログラ

ム間で制御転送データを送信するステップと、

を含む、請求項14記載の方法。

【請求項17】前記制御転送モード手段が情報の1ビットで前記プログラム制御の転送を示す、請求項14記載の方法。

【請求項18】前記第1のプロセッサ・アーキテクチャのアドレス範囲が前記第2のプロセッサ・アーキテクチャのアドレス範囲より小さい、請求項14記載の方法。

【請求項19】前記プログラム制御転送手段が前記第1のプロセッサ・アーキテクチャから前記第2のプロセッサ・アーキテクチャへの前記プログラム制御転送を示す場合、前記保持手段に前記第1のプロセッサ・アーキテクチャの基底アドレスのオフセットを保持するステップと、

前記プログラム制御転送手段が前記第2のプロセッサ・アーキテクチャから前記第1のプロセッサ・アーキテクチャへの前記プログラム制御転送を示す場合、前記保持手段に前記第2のプロセッサ・アーキテクチャのアドレス・ビットを保持するステップと、

を更に含む、請求項14記載の方法。

【請求項20】マイクロプロセッサと外部メモリとを有するマイクロプロセッサ・システムであって、前記マイクロプロセッサは、前記外部メモリ、レジスタ及び実行ユニット間で接続されるバス及びキャッシュ・ユニットとを有し、前記マイクロプロセッサは、自身が操作する前記レジスタにプロセス・スタックを格納するように構成され、前記マイクロプロセッサ・システムは、

a) 第1の連係方法に従って第1の連係データを格納するように構成される第1のマイクロプロセッサ・アーキテクチャと、

b) 第2の連係方法に従って第2の連係データを格納するように構成される第2のマイクロプロセッサ・アーキテクチャと、

c) 前記第1の連係データを前記第2の連係データに変換する手段と、

を有する、マイクロプロセッサ・システム。

【請求項21】前記第2の連係データを前記第1の連係データに変換する手段を更に含む、請求項20記載のマイクロプロセッサ・システム。

【請求項22】前記連係データが第1及び第2のマイク

ロプロセッサ・アーキテクチャの一方によってメモリに記憶されることが必要ない場合、前記第1及び第2のマイクロプロセッサ・アーキテクチャの他方からの前記連係データをレジスタに保持する手段を更に含む、請求項20記載のマイクロプロセッサ・システム。

【請求項23】前記第2のマイクロプロセッサ・アーキテクチャは前記第1の連係方法をエミュレートする方法を更に含む、請求項20記載のマイクロプロセッサ・システム。

【請求項24】プログラム制御を転送する手段を含み、前記転送する手段が制御転送モード・フィールドを含み、前記フィールドが、前記第1及び第2のプロセッサ・アーキテクチャのどちらが前記プログラム制御を有するかを示す、請求項20記載のマイクロプロセッサ・システム。

【請求項25】前記制御転送モード・フィールドは、前記マイクロプロセッサにより外部バスを介して前記外部メモリに書込まれる、請求項24記載のマイクロプロセッサ・システム。

【請求項26】前記制御転送モード・フィールドが情報の1ビットを含み、前記1ビットが、どのアーキテクチャが前記プログラム制御を有するかを示す、請求項24記載のマイクロプロセッサ・システム。

【請求項27】前記第2のマイクロプロセッサ・アーキテクチャのアドレス範囲が、前記第1のマイクロプロセッサ・アーキテクチャのアドレス範囲より大きい、請求項20記載のマイクロプロセッサ・システム。

【請求項28】前記第1のマイクロプロセッサ・アーキテクチャがX86 CISCアーキテクチャである、請求項20記載のマイクロプロセッサ・システム。

【請求項29】前記第2のマイクロプロセッサ・アーキテクチャがPowerPC RISCアーキテクチャである、請求項20記載のマイクロプロセッサ・システム。

【請求項30】複数のマイクロプロセッサ及びメイン・メモリを有するマイクロプロセッサ・システムであって、前記マイクロプロセッサは、前記メイン・メモリ、レジスタ及び実行ユニット間で接続されるバス及びキャッシュ・ユニットを有し、自身が操作する前記レジスタにプロセス・スタックを格納するように構成され、前記マイクロプロセッサ・システムは、

a) 第1のマイクロプロセッサ・アーキテクチャと、

b) 第2のマイクロプロセッサ・アーキテクチャと、

c) 前記第1及び第2のプロセッサ・アーキテクチャの、プログラム制御を解放する一方から獲得する他方に前記プログラム制御を転送する記憶手段と、

を含み、前記記憶手段が、前記プログラム制御を解放するプロセッサ・アーキテクチャからの情報を有し、該情報が、前記プログラム制御を獲得するプロセッサ・アーキテクチャによりアクセスされ得る、前記マイクロプロセッサ・システム。

【請求項31】前記記憶手段は、前記プログラム制御を獲得する前記プロセッサ・アーキテクチャによって前記情報がメモリに記憶されることを必要としない場合、少なくとも1つのレジスタに前記情報を記憶する、請求項30記載のマイクロプロセッサ・システム。

【請求項32】前記第2のアーキテクチャが前記第1のアーキテクチャをエミュレートする手段を更に有する、請求項30記載のマイクロプロセッサ・システム。

【請求項33】前記記憶手段は、前記第1及び第2のマイクロプロセッサ・アーキテクチャのどちらが前記プログラム制御を有するかを示す制御転送モード・フィールドを有する、請求項30記載のマイクロプロセッサ・システム。

【請求項34】前記メイン・メモリにある前記制御転送モード・フィールドが、前記複数のマイクロプロセッサの1つに転送される、請求項33記載のマイクロプロセッサ・システム。

【請求項35】前記制御転送モード・フィールドが情報の1ビットを含み、前記1ビットが、どのアーキテクチャがプログラム制御を有するかを示す、請求項33記載のマイクロプロセッサ・システム。

【請求項36】前記第2のマイクロプロセッサ・アーキテクチャのアドレス範囲が、前記第1のマイクロプロセッサ・アーキテクチャのアドレス範囲より大きい、請求項30記載のマイクロプロセッサ・システム。

【請求項37】前記第1のマイクロプロセッサ・アーキテクチャがX86 CISCアーキテクチャである、請求項30記載のマイクロプロセッサ・システム。

【請求項38】前記第2のマイクロプロセッサ・アーキテクチャがPowerPC RISCアーキテクチャである、請求項30記載のマイクロプロセッサ・システム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、一般にコンピュータ・システムの制御転送に関し、特に2つの異なるコンピュータ・アーキテクチャ間におけるコンピュータ・システムの制御転送に関する。

【0002】

【従来の技術】最新のコンピュータ・アーキテクチャの多くは、一般に旧式のコンピュータ・アーキテクチャよりも大きな処理能力を実現している。たとえば、RISCプロセッサ・アーキテクチャを例にとると、X86 CISCアーキテクチャを含むCISCプロセッサ・アーキテクチャに対して明確な性能優越性を有する実施形態を生み出した。しかしながら、X86アーキテクチャなどの旧式のアーキテクチャのために開発された膨大な量のソフトウェアは、市場の新たなプロセッサの導入に対して大きな障害となり、このソフトウェア・ベースを新たなアーキテクチャへ移行させるには莫大な時間と資金を要し、危険性が伴う。多数のRISCプロセッサ及

びシステムのベンダーは、ソフトウェアのX86アーキテクチャをエミュレートして、X86遺産のソフトウェアを現行製品に移行させようとしている。

【0003】ソフトウェア・エミュレーション技法においては重大な2つの欠点がある。1)ソフトウェア・エミュレーションはエミュレータ及びエミュレートされたプログラムを保持するのに必要なメモリにかなりの投資を必要とし、2)ソフトウェア・エミュレーションは本質的に速度が遅い。現在のソフトウェア・エミュレーション技術は、前の世代のX86プロセッサのロー・エンド並の性能しか発揮せず、一般に16メガバイトの物理的メモリを必要とする。

【0004】ソフトウェア・エミュレーションにおける速度の問題に対する解決策の1つは、2つ以上のマイクロコントローラを並列処理で使用するることである。米国特許番号第4370709号"Computer Emulator With Three Segment Microcode Memory And Two Separate Microcontrollers For Operand Derivation And Execution Phases"(1983年1月、Fosdickに付与、Tracor Inc.に譲渡)で開示される複数の内蔵型マイクロコントローラは、マクロ命令を2つのフェーズに分類することによって、より大きなマクロコントローラ命令をエミュレートする。1つのマイクロコントローラがマクロ命令を2つのフェーズに分類し、それから代行して2つの他のマイクロコントローラを制御して2つのフェーズのそれぞれを操作する。しかしながら、このシステムは、大型のコンピュータ・ハードウェアを必要とし、エミュレーション・プログラム及びエミュレータのために物理的なメモリにかなりの投資が必要とされるため、非常に高価である。

【0005】複数のアーキテクチャ間のソフトウェア・プログラムの変換或いはエミュレーションをサポートする処理システムの他の例では米国特許番号第5280589号"Memory Access Control System For Use With As Relatively Small Size Data Processing System"(1994年1月発行、株式会社東芝に譲渡)、同第4514803号"Methods For Partitioning Mainframe Instruction Sets To Implement Microprocessor Based Emulation Thereof"(1985年4月発行、IBMに譲渡)、同第5247520号"Communications Architecture Interface"(1993年9月発行、IBMに譲渡)等がある。上述の全特許が本発明に関連しているので参照されたい。

【0006】上述の開示されたコンピュータ・システムは、異なるアーキテクチャ間においてある種の通信が可能であるが、異なるワード長を有するアーキテクチャ間では通信はできない。PowerPCとX86の場合、X86アーキテクチャは現在32ビット・アーキテクチャとして定義されており、一方PowerPCは32ビット及び64ビット・アーキテクチャの両方である。このことは、完

全な制御転送を容易にするために32ビットX86コードを64ビットPowerPC空間に戻すことができなければならないことを意味する。更に、上述の特許群は、内部または外部メモリの実質的使用を必要とし、これによりプロセッサの処理速度が遅くなる。特にX86の互換性は、パラメータ及びプロセス戻り情報の転送のために、一般にメモリでプロセス・スタックが必要となる。プロセス・スタックの使用は、データをメモリに格納したり、またメモリからロードする必要があるため、しばしば処理が遅くなり、性能を落とすことになる。

【0007】

【発明が解決しようとする課題】従って本発明の目的は、異なるワード長及び連係方法（すなわちパラメータ、スタック・ポインタ及び戻りアドレスを送信する方法）を有するプロセッサ・アーキテクチャ間で、プログラム制御転送ができるコンピュータ・システムを提供することである。

【0008】本発明の別の目的は、パラメータ及びプロセス戻り情報を転送するためにメモリ内でプロセス・スタックの使用を必要としないコンピュータ・システムを提供することである。

【0009】本発明の別の目的は、制御転送に使用されるX86アプリケーション或いは他の同様なアプリケーションに対して逆方向の互換性を提供することである。

【0010】

【課題を解決するための手段】前述の及び本発明の他の目的は、複数のアーキテクチャ間においてプログラム制御を転送するマイクロプロセッサによって実現され、例えばRISCとX86CISCにおいて、記述子テーブル・エントリの1ビットを使用することにより、ターゲット戻りセグメントが第1のアーキテクチャまたは第2のアーキテクチャからのソフトウェアを含んでいるかどうかを示すことができる。第1のアーキテクチャの連係方法に使用されたが第2のアーキテクチャによって事前に指定されなかったレジスタを、第2のアーキテクチャによって使用して過剰のワード・ビットを保持することにより、両方のアーキテクチャが異なるワード長を使用できるようにする。更に、呼び出しプログラムのスタック・ポインタ及び戻りアドレスはメモリではなくレジスタに残され、そのため制御転送の性能が改良される。

【0011】

【発明の実施の形態】図1を参照すると、本発明のコンピュータ・システムの概要が示されている。中央処理ユニット（CPU）1はバス3を介してメイン・メモリ2と通信する。制御の転送中、CPU1はメイン・メモリ2と要素の読出しまたは書込みを行い、或いは後で説明するレジスタ9と読出しまたは書込みを実行する。CPU1は、バス及びキャッシュ・ユニット4、実行ユニット11、命令エミュレーション・ユニット5、及び命令ディスパッチ及び完了ユニット8を有する。命令エミュ

レーション・ユニット5は適切な位置で命令を取り出すための命令フェッチ・ブロック6、及び命令がデコードされる命令デコード・ブロック7を有する。レジスタ9と命令ディスパッチ・ブロック10は命令ディスパッチ及び完了ユニット8を形成する。

【0012】また、バス3はCPU1を第2のCPUに接続する。第2のCPUアーキテクチャは第1のCPUアーキテクチャとは異なるワード長、及び（または）パラメータ、スタック・ポインタ及び戻りアドレスの送信を含む異なる方法の呼び出しプログラムを有する。プロセッサ間にはマスタ／スレーブ関係が存在する。

【0013】図5乃至図7はX86アーキテクチャがどのようにインターレベル制御転送中、及びその前後においてプロセス・スタックの構造を指定するかを示す。特定の参照がIBMのPowerPC RISC及びX86 CISCプロセッサ・アーキテクチャに対して行われるが、当業者は2つのアーキテクチャ・プロセッサを設計する場合、本発明がこれらの特定の詳細及び他のプロセッサ・アーキテクチャなしで実行できることは当業者には容易に理解できよう。

【0014】制御転送が行われる前に、図5で示されるように呼び出しプログラムはパラメータ（最大32バイトまで）をそのプロセス・スタックに置くことになる。このスタックの位置はスタック・ポインタSS、ESPによって指定される。ここでSSはスタック・セクタ・レジスタであり、ESPはスタックの基底アドレスからスタックまでのオフセットを保持するレジスタである。スタックの基底アドレスはセクタSSによって参照されるスタックの記述子に含まれる。

【0015】図6で示されるように、X86プログラムが他のX86プログラムを呼び出す場合、幾つかの項目が呼び出されたプログラムのスタックにプッシュされる。最初に呼び出しプログラムのスタック・ポインタ（SS、ESP）が新たなスタックにプッシュされ、古いスタックからのパラメータが引き続く。最後に、呼び出しプログラムの戻りアドレス（CS、EIP）がスタックにプッシュされる。CSはコード・セクタ・レジスタであり、EIPはコード・セグメント基底アドレスからのオフセットをコード・セグメントに保持するレジスタである。基底アドレスはコード記述子に含まれ、セクタCSによって参照される。SS、ESPレジスタは新たなスタックの位置によって更新され、及びCS、EIPレジスタは呼び出されたプログラムのアドレスによって更新されることに注意されたい。

【0016】呼び出されたプログラムが呼び出し側に戻される場合、古いCS、EIPの値は新たなスタックからポップオフされ、戻りアドレスによってCS、EIPレジスタを更新するのに使用される。古いSS、ESP値はまた、新たなスタックからポップオフされ、最初に呼び出されたときのパラメータとして送信されたバイト

数だけ減らされて、SS、ESPレジスタを更新するのに使用される。図7は、戻り後の古いスタックの最終状態を示す。

【0017】この形態のパラメータ、スタック・ポインタ、及び戻りアドレスを送信するプロセスは、まとめてX86連係方法またはX86連係規則と呼ぶことができる。

【0018】PowerPCアーキテクチャは、プログラム制御転送において連係規則を明白に指定しないので、あるオペレーティング・システムは、仕事を遂行するのに必要な最適なものとして任意の規則を指定する可能性がある。しかしながら、人工遺物として、PowerPCからX86への制御転送には、呼び出しPowerPCプログラムがX86連係規則に従う必要がある。従ってそのような呼び出しは、古いスタックの存在に関係無く、直接呼び出されたX86プログラムにプロセス・スタックを構築するためにPowerPCプログラムを必要とする。技術的に、古いスタック・ポインタ(SS、ESP)は新たなプロセス・スタック内に置く必要はない。これはPowerPCアーキテクチャはSSまたはESPレジスタを指定しないからである。

【0019】古いSS、ESP値は、PowerPC呼び出しの場合スタックに置く必要はないので、本発明ではその位置の目的を再指定し、32ビットのX86ソフトウェアから64ビットのアドレス空間に戻る問題を解決する。再び図1を参照すると、呼び出しがPowerPCコードからX86コードに行われる場合、PowerPCコードは64ビットの戻りアドレスの上位32ビットを、X86とX86間の制御転送の古いESP用の予約位置に置く。戻りアドレスの下位32ビットは、古いEIP位置に置かれる。これらのアドレス要素は、CPUとメモリ間のバス3を介してスタックに置かれる。X86プログラムがPowerPCプログラムに戻る場合、プロセッサ・ハードウェアは、X86プロセス・スタックの位置から再びバス3を介して戻りアドレスの上位及び下位をポップすることになる。バス・ユニット4は2つの要素を命令エミュレーション・ユニット5に転送し、ここでその2つの要素を連結して単一の64ビット・アドレスを形成する。この64ビット・アドレスは命令フェッチ・ブロック6によって使用されて新たな位置で命令を取り出し始め、効果的に制御をPowerPCプログラムに転送して戻す。

【0020】図8はPowerPCプログラムがX86プログラムを呼び出した後の再指定されたプロセス・スタック(呼び出し後の"新たな"スタック)を示す。32ビットのPowerPCコードは戻りアドレスの上位32ビットに単純に0を書込むので、X86とPowerPCの32ビットのアドレス空間は互いにオーバーレイする。

【0021】上述したように本発明において、プロセッサ・ハードウェアは、2つのメカニズムのうちのどちら

を使用するかを、X86プログラムから呼び出しプログラムに戻る際に決めねばならない。実質的には、戻りプログラムがPowerPCかX86かであるかはソフトウェアによって告げられる。これはハードウェアがプログラムの戻りを実行する際に古いCSセクタ及び関連する記述子テーブル・エントリを調べることによって行われる。このようにX86アーキテクチャは、CSセクタがプロセス・スタックから読出される際、古いCSセクタ値がプロセッサによってロードされたメイン・メモリの記述子を参照して知ることができる。記述子テーブル・エントリは、CSセクタによって参照されたメモリのセグメントについての情報を有する。この記述子エントリの1ビット、特に記述子の上位4バイト(ダブルワード)のビット位置21は、32ビットX86アーキテクチャによって指定されないまま残る。本発明では、上位のダブルワードのビット位置21を、ターゲット戻りセグメントが標準のX86ソフトウェア(この時のビットは0)か、またはPowerPCソフトウェア(この時のビットは1)であるかどうかを示すものと定義する。ビット位置21は以降、制御転送モード・ビットとして示される。図9、図10及び図11は、X86アーキテクチャの3つの基本セグメント記述子、すなわち、データ・セグメント記述子、コード・セグメント記述子、及びシステムセグメント記述子のレイアウトを示す図である。また制御転送モード・ビット位置21も示されている。

【0022】従って、X86の戻りにより、メイン・メモリにある関連する記述子を読取るのと同様に、プロセッサ・ハードウェアがメイン・メモリのプロセス・スタックから古いCSセクタ値を読取る。命令エミュレーション・ユニット5(図1参照)は、発生した制御転送の形態を確かめるために、新たに指定された制御転送モード・ビットを調べる。制御転送モード・ビットが0であれば標準X86制御戻りが生じている。制御転送モード・ビットが1であれば、戻りPowerPCアドレスの上位及び下位の32ビットはプロセス・スタックから読取られ、フル64ビットの戻りアドレスを形成する。それから制御はこのアドレスにある命令に転送される。

【0023】この本発明の第2の態様は、X86プログラムからPowerPCプログラムへの制御転送に関する。PowerPCアーキテクチャは固定された連係規則を指定しないので、X86アーキテクチャの連係規則にも制限されない。X86プロセス・スタックはPowerPCプログラムに送信されるパラメータを有するが、パラメータ及び古いSS、ESP及びCS、EIP値を有するPowerPCプログラムに対して新たなスタックを構築する必要はない。実際にこれらの値を、メモリに書込むのではなくプロセッサ・レジスタに直接残すことにより、X86からPowerPCへの呼び出し性能は改良することができる。PowerPCからX86への戻りは、戻りアドレスを最初にメモリか

11

ら読取るよりも、レジスタに保持しレジスタの戻りアドレスに直接分岐することによって改良できる。

【0024】従って、本発明の第2の態様は、呼び出しX86プログラムのスタック・ポインタ及び戻りアドレスをメモリではなくレジスタに置くことである。これは呼び出されたプログラムのアーキテクチャ、この場合PowerPCであるが、X86アーキテクチャの連係制限に拘束されないからである。プロセッサはX86とX86間の制御転送と、X86とPowerPC間の制御転送とを識別できなければならない。この識別は、転送(X86のジャンプまたは呼び出し)を開始するときを使用される命令により参照されるゲート記述子をプロセッサが調べることによって達成される。このようにX86アーキテクチャは、ゲート記述子が、ターゲット・コード・セグメントを指すセクタを保持することがわかる。命令エミュレーション・ユニットは、ゲート記述子のセクタによって参照されたコード・セグメント記述子を読取るが、これらの両方はメイン・メモリにある。コード・セグメント記述子は命令エミュレーション・ユニットによって読取られた後、制御転送モード・ビットを参照して次に発生する転送の種類を決める。制御転送モード・ビットが0の場合、通常の制御転送が発生する。制御転送モード・ビットが1の場合、PowerPCコードへの転送が生じ、戻りアドレス及び古いスタック・ポインタはメモリではなくプロセッサ・レジスタに残る。この実施形態でこれらの値を保持するために指定されるレジスタが図12に示されている。

【0025】本発明は、特に好適な実施例に対して図示、説明されたが、当業者は本発明の趣旨、及び範囲内で形状及び詳細を様々に変更できることが理解できよう。

【0026】まとめとして、本発明の構成に関して以下の事項を開示する。

【0027】(1) a) 第1のマイクロプロセッサ・アーキテクチャと、
b) 前記第1のマイクロプロセッサ・アーキテクチャとは異なる第2のマイクロプロセッサ・アーキテクチャと、
c) 前記第1及び第2のマイクロプロセッサ・アーキテクチャの、プログラム制御を解放する一方から獲得する他方に該プログラム制御を転送する手段と、を含み、前記プログラム制御を転送する手段が、該プログラム制御を解放するマイクロプロセッサ・アーキテクチャから情報を受け取り、該プログラム制御を獲得するマイクロプロセッサ・アーキテクチャに前記情報を提供する、マイクロプロセッサ。

(2) 前記プログラム制御を転送する手段が、前記プログラム制御を獲得する前記マイクロプロセッサ・アーキテクチャが前記プログラム制御を解放する前記マイクロプロセッサ・アーキテクチャからのデータをメモリに必

12

要としない場合、レジスタに該データを保持する手段を含む、前記(1)記載のマイクロプロセッサ。

(3) 前記データを保持する手段が、前記プログラム制御を解放する前記マイクロプロセッサ・アーキテクチャの基底アドレスのオフセットを含む、前記(2)記載のマイクロプロセッサ。

(4) 前記第2のマイクロプロセッサ・アーキテクチャが、前記第1のマイクロプロセッサ・アーキテクチャをエミュレートする手段を更に含む、前記(1)記載のマイクロプロセッサ。

(5) 前記プログラム制御を転送する手段は、前記第1及び第2のマイクロプロセッサ・アーキテクチャのどちらが前記プログラム制御を有するかを示す情報フィールドを含む、前記(1)記載のマイクロプロセッサ。

(6) 外部メモリにある前記情報フィールドが、バスによって前記マイクロプロセッサに転送される、前記(5)記載のマイクロプロセッサ。

(7) 前記情報フィールドが情報の1ビットを含み、前記ビットの状態が、前記第1及び第2のマイクロプロセッサ・アーキテクチャのどちらが前記プログラム制御を有するかを示す、前記(5)記載のマイクロプロセッサ。

(8) 前記第2のマイクロプロセッサ・アーキテクチャのアドレス範囲が、前記第1のマイクロプロセッサ・アーキテクチャのアドレス範囲より大きい、前記(1)記載のマイクロプロセッサ。

(9) 前記第1のマイクロプロセッサ・アーキテクチャはX86 CISCアーキテクチャである、前記(1)記載のマイクロプロセッサ。

(10) 前記第2のマイクロプロセッサ・アーキテクチャはPowerPC RISCアーキテクチャである、前記(1)記載のマイクロプロセッサ。

(11) 第1及び第2のプロセッサ・アーキテクチャ間でプログラム制御を転送する方法であって、

a) 前記第1のプロセッサ・アーキテクチャに対応する第1のプログラムを与えるステップと、

b) 前記第2のプロセッサ・アーキテクチャに対応する第2のプログラムを与えるステップと、

c) 前記第1及び第2のプロセッサ・アーキテクチャを示す複数の状態を有する制御転送モード手段を与えるステップと、

d) 次に実行される命令を有する前記プログラムに対応するプロセッサ・アーキテクチャを示す前記複数の状態の1つに、前記制御転送モード手段を設定するステップと、

e) 制御転送データをメイン・メモリまたはハードウェア・レジスタのどちらに送信するかを決めるために、前記制御転送モード手段の状態を使用するステップと、

f) 前記制御転送モード手段の状態に従って、前記第1及び第2のプログラム間で制御転送データを送信するス

テップと、を有する方法。

(12) 前記制御転送モード手段は、情報の単一ビットの値にもとづいて前記第1及び第2のプロセッサ・アーキテクチャを示す、前記(11)記載の方法。

(13) 前記第1のプロセッサ・アーキテクチャのアドレス範囲が前記第2のプロセッサ・アーキテクチャのアドレス範囲より小さい、前記(11)記載の方法。

(14) マイクロプロセッサ、及び外部バスで前記マイクロプロセッサに接続されている外部メモリを有するマイクロプロセッサ・システムの第1及び第2のプロセッサ・アーキテクチャ間にてプログラム制御を転送する方法であって、前記マイクロプロセッサは、前記外部メモリ、レジスタ及び実行ユニット間で接続されるバス及びキャッシュ・ユニットを有し、前記マイクロプロセッサは、自身が操作する前記レジスタにプロセス・スタックを格納するように構成され、前記方法は、

a) 制御転送モード手段にて、前記第2のプロセッサ・アーキテクチャから前記第1のプロセッサ・アーキテクチャへの前記プログラム制御の転送を示すステップと、
b) 保持手段において前記第2のプロセッサ・アーキテクチャに関するデータを保持するステップと、
c) 前記プログラム制御を前記第2のプロセッサ・アーキテクチャから前記第1のプロセッサ・アーキテクチャに転送するステップと、
d) 前記第2のプロセッサ・アーキテクチャに関するデータを、前記プログラム制御の転送の際に前記保持手段から前記第2のプロセッサ・アーキテクチャに戻すステップと、を含む方法。

(15) e) 前記制御転送モード手段を、前記外部バスを介して前記外部メモリに適用するステップと、を含む、前記(14)記載の方法。

(16) 前記プログラム制御を前記第2のプロセッサ・アーキテクチャから前記第1のプロセッサ・アーキテクチャに転送するステップが、

c1) プログラムのアーキテクチャに従って、前記制御転送モード手段の状態を設定するステップと、
c2) 制御転送データが外部メモリまたはハードウェア・レジスタのどちらに送信されるかを定めるために前記制御転送モード手段の状態を使用するステップと、
c3) 前記制御転送手段の前記状態に従って、プログラム間で制御転送データを送信するステップと、を含む、前記(14)記載の方法。

(17) 前記制御転送モード手段が情報の1ビットで前記プログラム制御の転送を示す、前記(14)記載の方法。

(18) 前記第1のプロセッサ・アーキテクチャのアドレス範囲が前記第2のプロセッサ・アーキテクチャのアドレス範囲より小さい、前記(14)記載の方法。

(19) 前記プログラム制御転送手段が前記第1のプロセッサ・アーキテクチャから前記第2のプロセッサ・ア

ーキテクチャへの前記プログラム制御転送を示す場合、前記保持手段に前記第1のプロセッサ・アーキテクチャの基底アドレスのオフセットを保持するステップと、前記プログラム制御転送手段が前記第2のプロセッサ・アーキテクチャから前記第1のプロセッサ・アーキテクチャへの前記プログラム制御転送を示す場合、前記保持手段に前記第2のプロセッサ・アーキテクチャのアドレス・ビットを保持するステップと、を更に含む、前記(14)記載の方法。

(20) マイクロプロセッサと外部メモリとを有するマイクロプロセッサ・システムであって、前記マイクロプロセッサは、前記外部メモリ、レジスタ及び実行ユニット間で接続されるバス及びキャッシュ・ユニットとを有し、前記マイクロプロセッサは、自身が操作する前記レジスタにプロセス・スタックを格納するように構成され、前記マイクロプロセッサ・システムは、

a) 第1の連係方法に従って第1の連係データを格納するように構成される第1のマイクロプロセッサ・アーキテクチャと、
b) 第2の連係方法に従って第2の連係データを格納するように構成される第2のマイクロプロセッサ・アーキテクチャと、
c) 前記第1の連係データを前記第2の連係データに変換する手段と、を有する、マイクロプロセッサ・システム。

(21) 前記第2の連係データを前記第1の連係データに変換する手段を更に含む、前記(20)記載のマイクロプロセッサ・システム。

(22) 前記連係データが第1及び第2のマイクロプロセッサ・アーキテクチャの一方によってメモリに記憶されることが必要ない場合、前記第1及び第2のマイクロプロセッサ・アーキテクチャの他方からの前記連係データをレジスタに保持する手段を更に含む、前記(20)記載のマイクロプロセッサ・システム。

(23) 前記第2のマイクロプロセッサ・アーキテクチャは前記第1の連係方法をエミュレートする方法を更に含む、前記(20)記載のマイクロプロセッサ・システム。

(24) プログラム制御を転送する手段を含み、前記転送する手段が制御転送モード・フィールドを含み、前記フィールドが、前記第1及び第2のプロセッサ・アーキテクチャのどちらが前記プログラム制御を有するかを示す、前記(20)記載のマイクロプロセッサ・システム。

(25) 前記制御転送モード・フィールドは、前記マイクロプロセッサにより外部バスを介して前記外部メモリに書込まれる、前記(24)記載のマイクロプロセッサ・システム。

(26) 前記制御転送モード・フィールドが情報の1ビットを含み、前記1ビットが、どのアーキテクチャが前

記プログラム制御を有するかを示す、前記(24)記載のマイクロプロセッサ・システム。

(27) 前記第2のマイクロプロセッサ・アーキテクチャのアドレス範囲が、前記第1のマイクロプロセッサ・アーキテクチャのアドレス範囲より大きい、前記(20)記載のマイクロプロセッサ・システム。

(28) 前記第1のマイクロプロセッサ・アーキテクチャがX86 CISCアーキテクチャである、前記(20)記載のマイクロプロセッサ・システム。

(29) 前記第2のマイクロプロセッサ・アーキテクチャがPowerPC RISCアーキテクチャである、前記(20)記載のマイクロプロセッサ・システム。

(30) 複数のマイクロプロセッサ及びメイン・メモリを有するマイクロプロセッサ・システムであって、前記マイクロプロセッサは、前記メイン・メモリ、レジスタ及び実行ユニット間で接続されるバス及びキャッシュ・ユニットを有し、自身が操作する前記レジスタにプロセス・スタックを格納するように構成され、前記マイクロプロセッサ・システムは、

a) 第1のマイクロプロセッサ・アーキテクチャと、
b) 第2のマイクロプロセッサ・アーキテクチャと、
c) 前記第1及び第2のプロセッサ・アーキテクチャの、プログラム制御を解放する一方から獲得する他方に前記プログラム制御を転送する記憶手段と、を含み、前記記憶手段が、前記プログラム制御を解放するプロセッサ・アーキテクチャからの情報を有し、該情報が、前記プログラム制御を獲得するプロセッサ・アーキテクチャによりアクセスされ得る、前記マイクロプロセッサ・システム。

(31) 前記記憶手段は、前記プログラム制御を獲得する前記プロセッサ・アーキテクチャによって前記情報がメモリに記憶されることを必要としない場合、少なくとも1つのレジスタに前記情報を記憶する、前記(30)記載のマイクロプロセッサ・システム。

(32) 前記第2のアーキテクチャが前記第1のアーキテクチャをエミュレートする手段を更に有する、前記(30)記載のマイクロプロセッサ・システム。

(33) 前記記憶手段は、前記第1及び第2のマイクロプロセッサ・アーキテクチャのどちらが前記プログラム制御を有するかを示す制御転送モード・フィールドを有する、前記(30)記載のマイクロプロセッサ・システム。

(34) 前記メイン・メモリにある前記制御転送モード・フィールドが、前記複数のマイクロプロセッサの1つに転送される、前記(33)記載のマイクロプロセッサ

・システム。

(35) 前記制御転送モード・フィールドが情報の1ビットを含み、前記1ビットが、どのアーキテクチャがプログラム制御を有するかを示す、前記(33)記載のマイクロプロセッサ・システム。

(36) 前記第2のマイクロプロセッサ・アーキテクチャのアドレス範囲が、前記第1のマイクロプロセッサ・アーキテクチャのアドレス範囲より大きい、前記(30)記載のマイクロプロセッサ・システム。

(37) 前記第1のマイクロプロセッサ・アーキテクチャがX86 CISCアーキテクチャである、前記(30)記載のマイクロプロセッサ・システム。

(38) 前記第2のマイクロプロセッサ・アーキテクチャがPowerPC RISCアーキテクチャである、前記(30)記載のマイクロプロセッサ・システム。

【図面の簡単な説明】

【図1】本発明のコンピュータ・システムを示すブロック図である。

【図2】本発明のバスに接続された異なるアーキテクチャのCPUを示す図である。

【図3】従来技術のシステム概念図である。

【図4】本発明のシステム概念図である。

【図5】本発明のプロセス・スタックを示す図である。

【図6】本発明のプロセス・スタックを示す図である。

【図7】本発明のプロセス・スタックを示す図である。

【図8】本発明の再指定されたプロセス・スタックを説明する図である。

【図9】本発明の識別情報を示す図である。

【図10】本発明の識別情報を示す図である。

【図11】本発明の識別情報を示す図である。

【図12】本発明のレジスタを示す図である。

【図13】スタックの位置とレジスタの関係を示す図である。

【符号の説明】

1 CPU

2 メイン・メモリ

3 バス

4 バス及びキャッシュ・ユニット

5 命令エミュレーション・ユニット

6 命令フェッチ・ブロック

7 命令デコード・ブロック

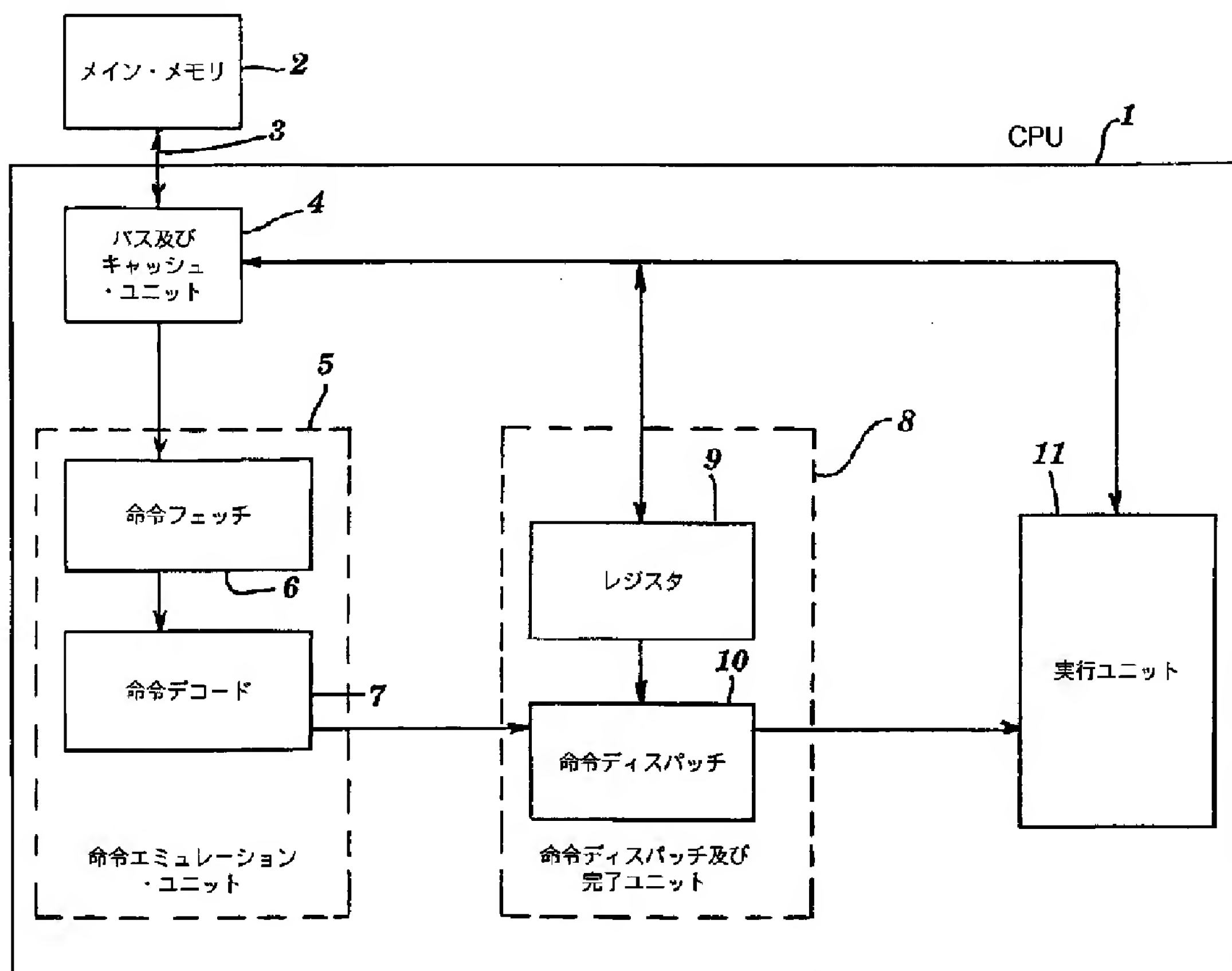
8 命令ディスパッチ及び完了ユニット

9 レジスタ

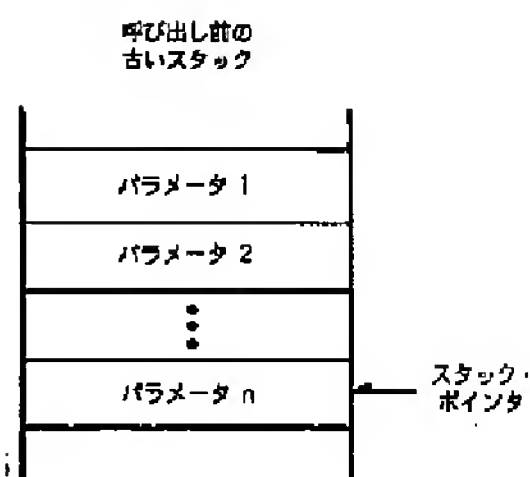
10 命令ディスパッチ・ブロック

11 実行ユニット

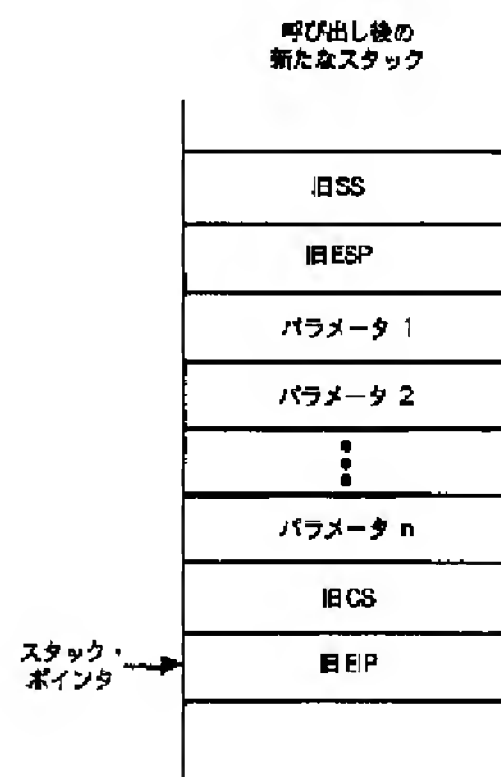
【図1】



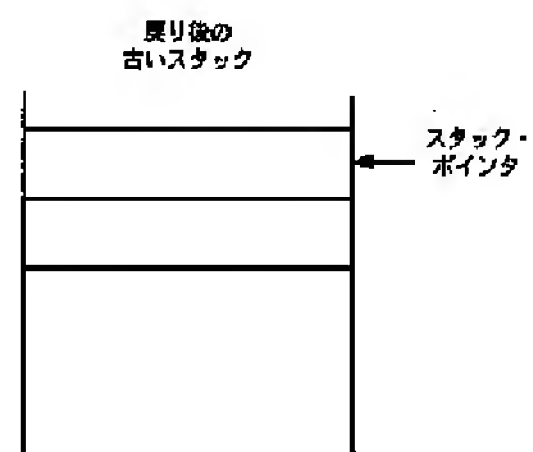
【図5】



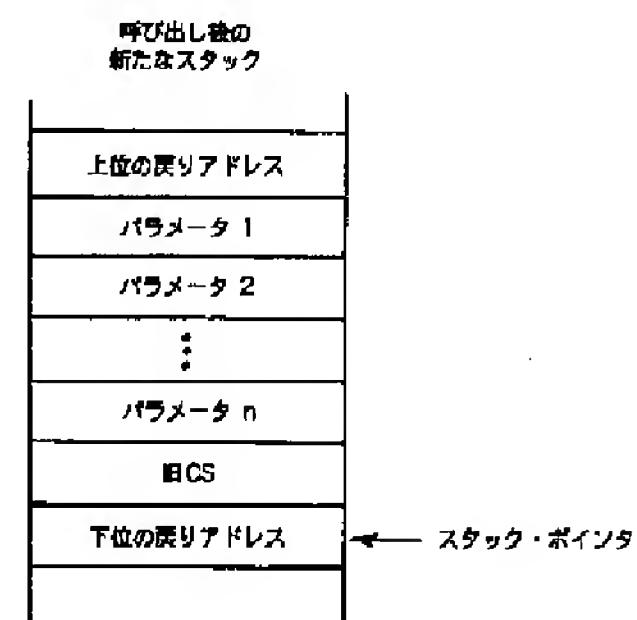
【図6】



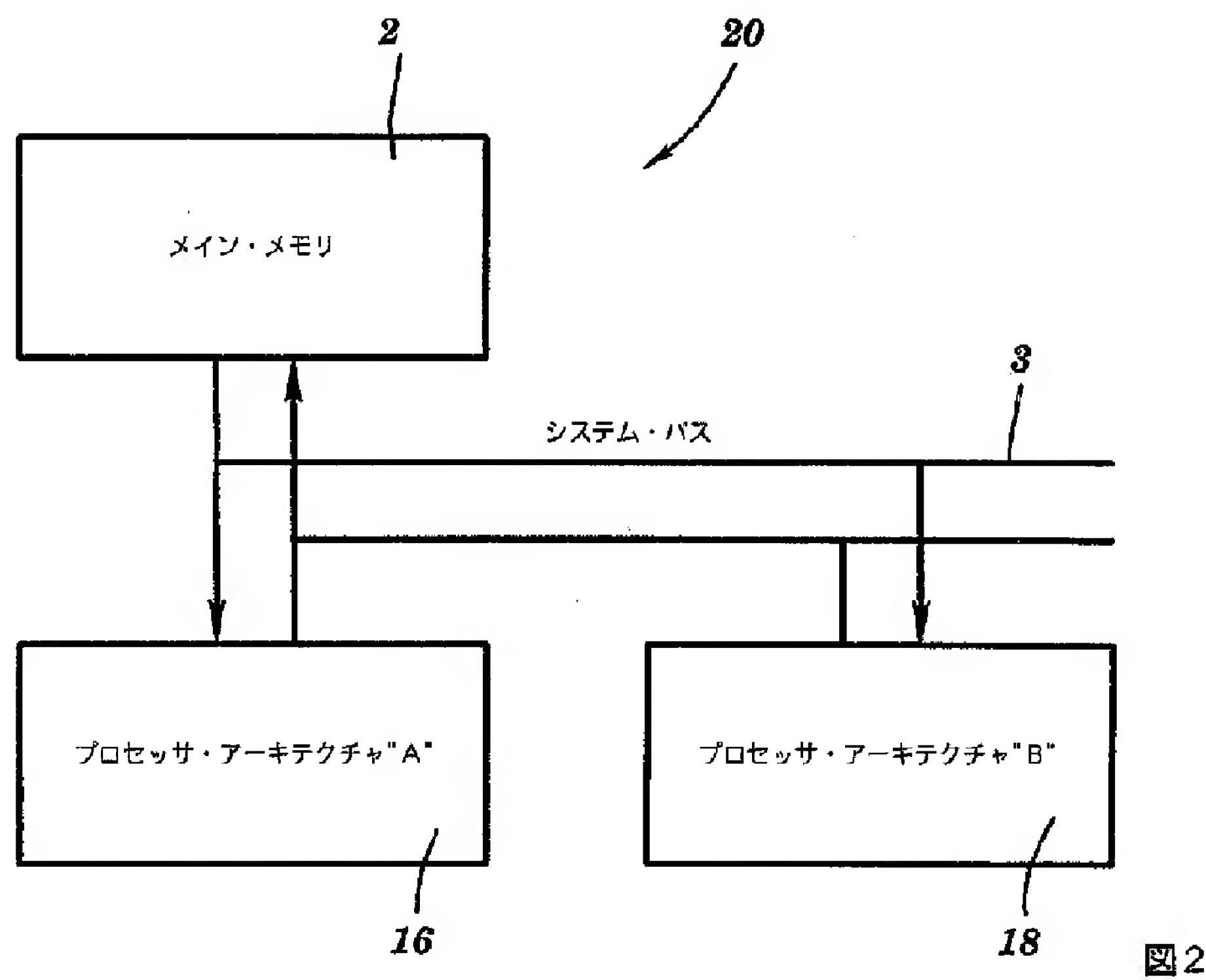
【図7】



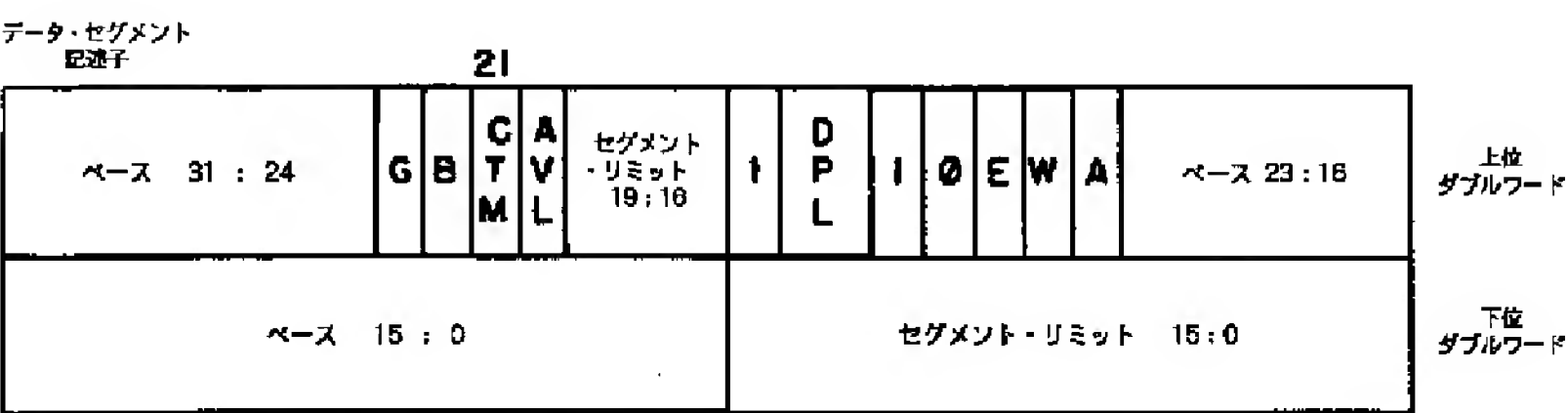
【図8】



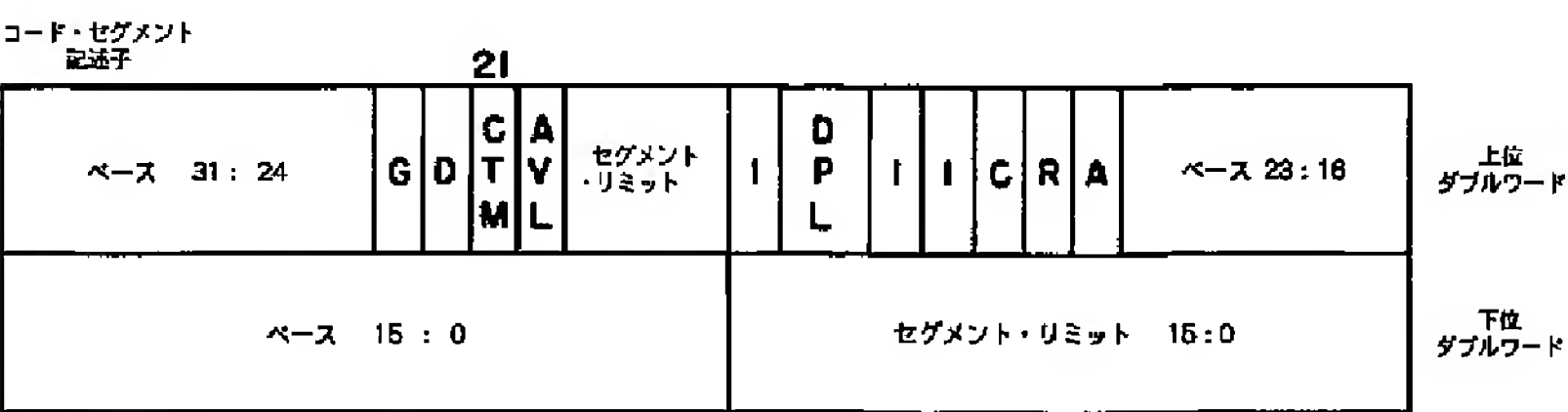
【図2】



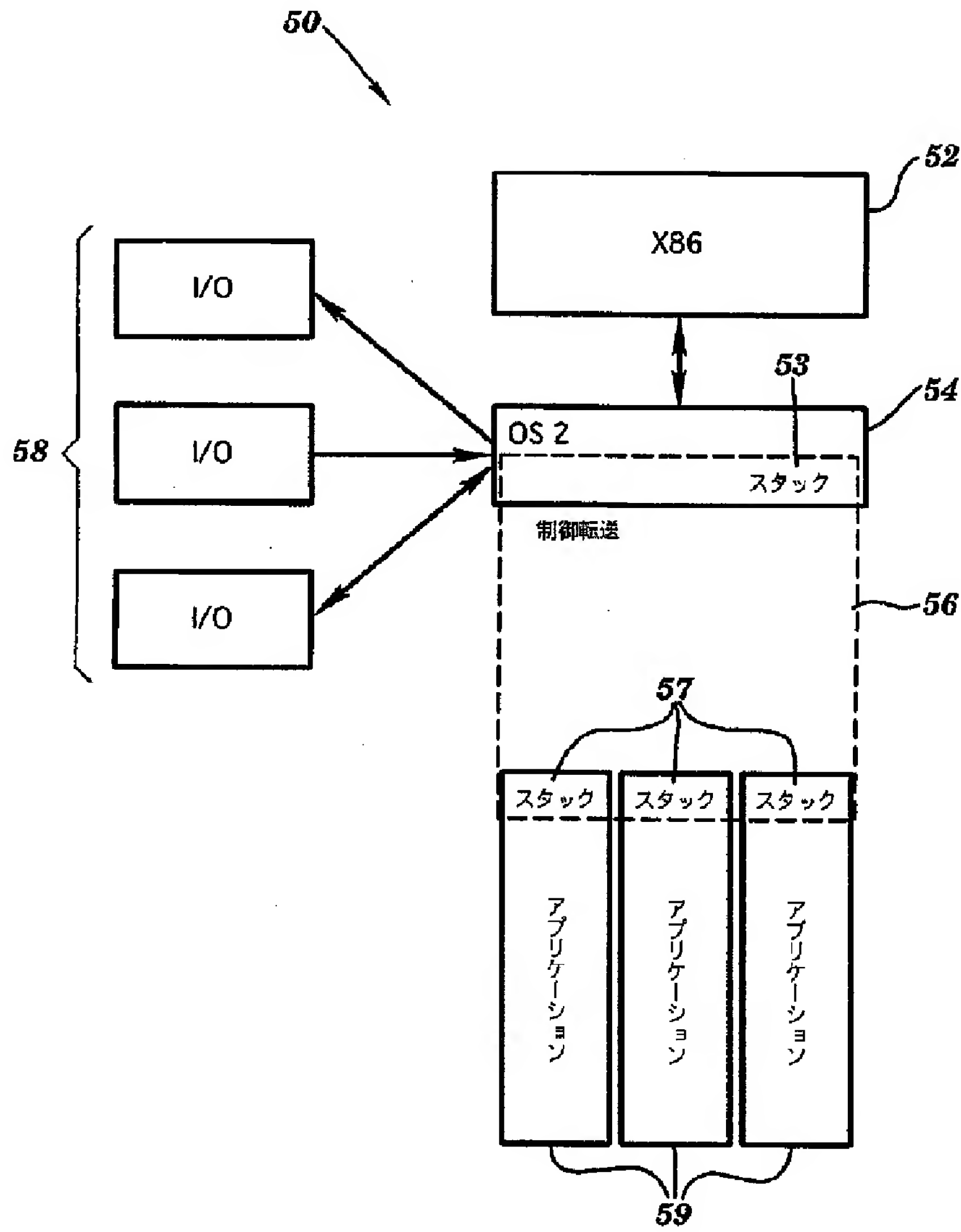
【図9】



【図10】



【図3】



【図11】

システム・セグメント
記述子

ベース 31 : 24	G	CTM	セグメント・リミット 19 : 18	I	DPL	0	タイプ	ベース 23 : 18	上位 ダブルワード
ベース 15 : 0				セグメント・リミット 15 : 0					下位 ダブルワード

【図4】

